On the Interaction between Load Balancing and Speed Scaling

Lijun Chen, Na Li and Steven H. Low
Engineering & Applied Science Division, California Institute of Technology, USA

***Abstract*— Speed scaling has been widely adopted in computer and communication systems, in particular, to reduce energy consumption. An important question is how speed scaling interacts with other resource allocation mechanisms such as scheduling and routing, etc. In this paper, we study the interaction of speed scaling with load balancing. We characterize the equilibrium resulting from the load balancing and speed scaling interaction, and introduce two optimal load balancing designs, in terms of traditional performance metric and cost-aware (in particular, energy-aware) performance metric respectively. Especially, we characterize the load-balancing-speed-scaling equilibrium with respect to the optimal load balancing schemes in processor sharing systems. Our results show that the degree of inefficiency at the equilibrium is mostly bounded by the heterogeneity of the system, but independent of the number of the servers. These results provide insights in understanding the interaction of load balancing with speed scaling and guiding new designs.**

***Index Terms*— Load balancing, Speed scaling, Energy efficiency, Efficiency loss, Data centers.**

I. Introduction

The energy consumption rate of computer and communication systems has been increasing exponentially. Computer and communication systems must make a fundamental tradeoff between performance and energy usage, see, e.g., [1], [2]. The addition of energy to standard performance metrics such as delay, throughput and loss fundamentally changes the problem space of some of resource allocation designs. Not only are new mechanisms needed to optimize energy usage, existing algorithms and protocols must be re-examined as a formerly optimal algorithm may now perform poorly with respect to a new energy-aware metric. Energy management decisions must be decomposed and coordinated spatially as well as temporally, and yet global optimality must be achieved through local algorithms that are implementable in a distributed manner. In this paper we study load balancing and its interaction with speed scaling.

Energy-aware speed scaling – to adapt the speed of the system so as to balance energy and performance metrics – is a widely-adopted power management technique, see, e.g., [3], [4], [5], [6], [7], [8], [9], [10]. Previous works on speed scaling usually focus on a single server and study its interaction with scheduling, see, e.g, [11], [8], [9], [10]. Here we consider a network setting and study the interaction of speed scaling with load balancing, to provide insights into such issues as: i) How does the system perform under speed scaling in terms of traditional performance metrics as well as energy-aware metrics? ii) How to design energy-aware optimal load balancing and can we decouple the design of load balancing from that of speed scaling? iii) How does the sophistication of speed scaling impact the design and performance of load balancing? We focus on gated-static speed scaling in processor sharing systems, and our results provide useful insights into the first two questions.

Specifically, we characterize the equilibrium resulting from the load balancing and speed scaling interaction, and introduce two optimal load balancing design problems, in terms of traditional performance metric and cost-aware (in particular, energy-aware) performance metric respectively. We study in detail the load-balancing-speed-scaling equilibrium and the optimal load balancing designs in processor sharing systems with gated-static speed scaling, and propose distributed load balancing algorithms to achieve the corresponding equilibrium and optima. Especially, we characterize the degree of inefficiency at the load-balancing-speed-scaling equilibrium, in terms of delay as well as energy-aware metric. We show that the degree of inefficiency is mostly bounded by the heterogeneity of the system, but independent of the number of servers in the system. Our results suggest that, as in many applications a low-order polynomial provides a good approximation to power function, we can decouple the design of load balancing from speed scaling without incurring much inefficiency in delay. In terms of power-aware performance metric, our results suggest that, as long as the heterogeneity in the system is small, we can decouple the design of load balancing from speed scaling without incurring much efficiency loss; but when the heterogeneity in the system is large, we have to do energy-aware load balancing if the energy consumption is a main concern.

The paper is organized as follows. The next section briefly discusses some related work. Section III describes the system model. Section IV gives a brief characterization of the load-balancing-speed-scaling equilibrium, and introduces two optimal load balancing design problems. Section V studies in detail the load-balancing-speed-scaling interaction in processor sharing systems with gated-static speed scaling. Section VI provides numerical examples to complement the theoretical analysis, and Section VII concludes with some discussion on further research.

II. Related work

Power management techniques have been increasingly adopted in designs from single-device level such as chips to network level such as data centers. It has spurred a new branch of research in its own right. In particular, starting with Yao et al [12], there is extensive research on analytical study of speed scaling, see, e.g., [13], [14], [15], [16], [17], [18], [11], [19], [8], [20], [21], [9], [10]. Bansal et al [8] show that a speed scaling policy (SRPT, $P^{-1}(n+1)$) is 3-competitive for regular power functions in the worst-case analysis. This result has been tighten and extended to PS scheduling as well as to stochastic analysis by Andrew et al [10]. Especially, Andrew et al [10] provide a comprehensive study of speed scaling and its interaction with scheduling, and show a fundamental tradeoff between optimality, fairness and robustness in speed scaling designs.

Related work also includes [22], [23] that show that the degree of inefficiency in delay for load balancing in processor sharing systems with fixed server speeds scales with the number of servers in the system. This result has been extended to the processor sharing system with multi-class load [24], and to other scheduling policies such as SRPT [25]. In contrast to these results, we show that the degree of inefficiency in delay for load balancing in processor sharing systems with speed scaling is bounded by the heterogeneity of the system, but independent of the number of servers.

III. System model

Consider a system with a set N of servers[1] and a Poisson arrival process of rate $\lambda > 0$; see Figure 1. We assume that job size is i.i.d., and without loss of generality, has a mean of 1. Associated with each server i is a service rate (or speed) s_i. There is a load balancing dispatcher that probabilistically routes arrivals to severs according to certain "traditional" performance metric $\mathcal{F}_i$ that end users are concerned with, so that $\mathcal{F}_i$ at each server i is the same and minimal. The metric $\mathcal{F}_i$ can be, for example, the mean response time $E[T_i]$ at the server, the summation of $E[T_i]$ and propagation delay τ_i, and the blocking probability p_i, etc.

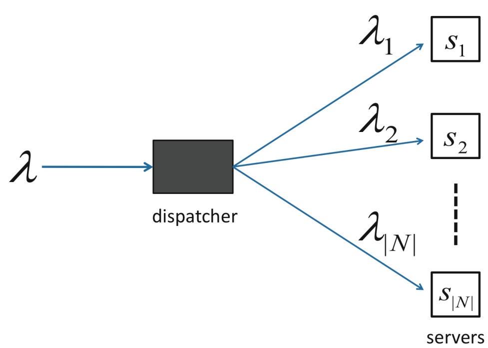

Fig. 1. A pictorial diagram of the system model.

It follows that the resulting arrival process to server i is Poisson with rate λ_i. We assume that server i's performance curve $\mathcal{F}_i = f_i(s_i, \lambda_i)$ (or its analytical approximation) is continuously differentiable, increasing in arrival rate λ_i, and decreasing in service capacity s_i with $f_i(\infty, \lambda) = 0$. This is a rather general assumption. In order to ensure stability, we must have $\lambda_i < s_i$ for all $i \in N$. We can thus assume that $f_i(s_i, \lambda_i) = \infty$ when $\lambda_i \geq s_i$.

Besides performance metric $\mathcal{F}_i$ that is perceived by end users, each server i incurs certain cost $c_i(s_i)$ per unit time when it runs at a speed of s_i. The cost can be, for example, the power expended at the server, or any other types of service costs. Given an incoming rate of λ_i, let $g_i(s_i, \lambda_i) = E[c_i(s_i)]$, the average cost. The average cost depends on the speed as well as the scheduling policy at the server. The cost function $g_i(s_i, \lambda_i)$ (or its analytical approximation) is assumed to be continuously differentiable, increasing in s_i, and non-decreasing in λ_i. Given arrival rate λ_i and scheduling policy, each server i will choose a speed s_i to minimize a "cost-aware" performance metric $\mathcal{M}_i$:

$$\mathcal{M}_i = g_i(s_i, \lambda_i) + \beta_i \lambda_i f_i(s_i, \lambda_i), \tag{1}$$

where $\beta_i > 0$ is used to characterize the relative weight of internal cost and traditional performance metric.

By the above model, we have actually assumed some kind of static speed scaling, i.e., choose a single speed s_i for a given arrival rate λ_i. With more complicated notation, we can also model dynamic scaling, i.e., adapt speed to different states such as the number of jobs in the server.

Speed scaling can be broadly defined as any behavior of adapting speed to load, and can be due to various reasons, corresponding to different choices of cost function $g_i(s_i, \lambda_i)$. In this paper, we will mostly focus on energy-aware speed scaling as a concrete system to study the interaction between load balancing and speed scaling, and consider the following performance metric:

$$\mathcal{M}_i = E[P_i(s_i)] + \beta_i \lambda_i E[T_i], \tag{2}$$

where $P_i(s_i)$ is the power expended when server i runs at speed s_i. The modeling of the power function $P_i(s_i)$ is an active research topic, and measurements have shown it can take on different forms depending on the system involved. In many applications a low-order polynomial form

$$P_i(s_i) = k_i s_i^{\alpha_i}, \quad k_i > 0, \quad \alpha_i > 1 \tag{3}$$

provides a good approximation. For example, for dynamic power in CMOS P_i is often assumed to be cubic in previous works [2]. We will focus on polynomial power function (3) in this paper, as in many previous works on speed scaling.

IV. Load-balancing-speed-scaling interaction

In this section, we characterize the equilibrium resulting from the interaction between load balancing and speed scaling for the general model described in Section III. We then introduce two optimal load balancing problems, $\mathcal{F}$-optimal load balancing and cost-aware optimal load balancing, under speed scaling. We intend to characterize the equilibrium with respect to those two optimal load balancing problems, as well

[1]Here a server can be a single server, or represent a cluster of collocated servers in, e.g., a micro-datacenter.

as proposing distributed load balancing algorithms to achieve the corresponding equilibrium and optima.

Given server speeds $(s_i)_{i\in N}$ and denote the set of servers used at load balancing by N_b, i.e., $i \in N_b$ iff $\lambda_i > 0$. At load balancing, the $\mathcal{F}_i$ value at any server $i \in N_b$ is thus the same, and not larger than the $\mathcal{F}_j$ value a job would experience if routed to any unused server $j \in N/N_b$. This can be written mathematically as

$$f_i(s_i, \lambda_i) \leq f_j(s_j, \lambda_j),\ \forall j \in N,\ \forall i \in N_b, \quad (4)$$
$$\sum_{i\in N} \lambda_i = \lambda, \quad (5)$$

where $(\lambda_i)_{i\in N}$ is the arrival rates at the servers at load balancing.[2] Denote the $\mathcal{F}_i$ value at server $i \in N_b$ at load balancing by γ. The load balancing condition (4)-(5) can be equivalently written as: there exists a $\gamma > 0$, such that

$$(f_i(s_i, \lambda_i) - \gamma)(\bar{\lambda}_i - \lambda_i) \geq 0,\ \forall\ \bar{\lambda}_i \geq 0, \quad (6)$$
$$\sum_{i\in N} \lambda_i = \lambda. \quad (7)$$

To see this equivalence, note that equations (6)-(7) imply that γ must equal the $\mathcal{F}_i$ value at server $i \in N_b$ at load balancing.

Assume that speed scaling problem $\min_{s_i > \lambda_i} \mathcal{M}_i$ has a unique solution $s_i(\lambda_i)$. Under the aforementioned assumptions on f_i and g_i, speed scaling $s_i(\lambda_i)$ satisfies:[3]

$$\frac{\partial g_i(s_i, \lambda_i)}{\partial s_i} + \beta_i \lambda_i \frac{\partial f_i(s_i, \lambda_i)}{\partial s_i} = 0. \quad (8)$$

Definition 1: The load-balancing-speed-scaling (LBSS) equilibrium is defined as a triple $\{(\lambda_i)_{i\in N}, (s_i)_{i\in N}, \gamma\}$ that satisfies the variational inequalities (6), (7) and (8).

The performance of the system under load balancing and speed scaling is determined by the LBSS equilibrium. At the LBSS equilibrium $\{(\lambda_i)_{i\in N}, (s_i)_{i\in N}, \gamma\}$, $s_i = s_i(\lambda_i)$ and

$$(f_i(s_i(\lambda_i), \lambda_i) - \gamma)(\bar{\lambda}_i - \lambda_i) \geq 0,\ \forall\ \bar{\lambda}_i \geq 0, \quad (9)$$
$$\sum_{i\in N} \lambda_i = \lambda. \quad (10)$$

The following result is straightforward [26].

Theorem 2: The LBSS equilibrium satisfies the local optimality condition for the following optimization problem:

$$\min_{\lambda_i \geq 0} \quad \sum_i \int f_i(s_i(\lambda_i), \lambda_i) d\lambda_i \quad (11)$$
$$\text{s.t.} \quad \sum_i \lambda_i = \lambda, \quad (12)$$

and $-\gamma$ is the corresponding optimal dual variable.

Proof: Note that LBSS equilibrium condition (9)-(10) is a variational inequality characterization of optimality condition for optimization problem (11)-(12) and its dual [26]. ■

[2]Note that in this paper we reload the notation, and λ_i denotes both any arrival rate of sever i and the arrival rate of server i at load balancing, depending on the context.

[3]The dynamic speed range of a server is usually finite, i.e., $s_i \leq r_i$ for some $r_i > 0$. For simplicity, we do not consider such a constraint in this paper. However, such a constraint does not change the general structure of our model, in terms of, e.g., equilibrium characterization, and distributed decomposition structure, etc.

An optimization problem characterization of the equilibrium is usually very useful. It captures the global structure of the problem, and often we can easily tell from the optimization problem if there exists an equilibrium, the multiplicity of the equilibria, as well as derive distributed or efficient algorithm to the equilibrium.

When there is no speed scaling, i.e., s_i is fixed, we recover the optimization problem characterization of usual load balancing. Under this situation, problem (11)-(12) is strictly convex as $f_i(s_i, \lambda_i)$ is an increasing function of λ_i, and the equilibrium is unique. In general, there may be no or multiple LBSS equilibria, depending on properties of performance curve $f_i(s_i(\lambda_i), \lambda_i)$ under speed scaling. For example, consider performance metric (2) with power function (3) in a processor sharing system with gated static speed scaling (see the next section). Speed scaling $s_i(\lambda_i)$ satisfies

$$\frac{\beta_i}{(s_i - \lambda_i)^2} = k_i(\alpha_i - 1)s_i^{\alpha_i - 2}.$$

When $\alpha_i < 2$, $f_i(s_i(\lambda_i), \lambda_i)$ is decreasing. So, problem (11)-(12) becomes a problem of minimizing a concave objective function, which is usually a hard computing problem and may admit multiple solutions.

In the above load balancing model, the dispatcher routes the arrivals according to "traditional" performance metric $\mathcal{F}_i$ but does not consider the internal cost g_i of the server. We call this model *cost-oblivious* load balancing (e.g., energy-oblivious in the case of energy-aware speed scaling). It can also be seen as a selfish routing game where each job chooses a server with minimal $\mathcal{F}_i$ value [27]. So, the LBSS equilibrium might not be socially optimal, in terms of metric $\mathcal{F}_i$ as well as energy-aware metric $\mathcal{M}_i$. As we mentioned before, speed scaling brings additional dimension such as energy into the design objective. It is of significant value to study its interaction with the existing algorithms and protocols, e.g., if it is optimal with respect to traditional performance metric $\mathcal{F}_i$ as well as a new one $\mathcal{M}_i$, how to design distributed optimal algorithms in terms of new performance metric, and if we can decouple speed scaling from other resource allocation mechanisms. In order to study these questions for load balancing, we consider two new load balancing models, as follows.

$\mathcal{F}$-optimal load balancing: The dispatcher routes arrivals so as to achieve social optimum in terms of traditional performance metric $\mathcal{F}_i$:

$$\min_{\lambda_i \geq 0} \quad \sum_i \lambda_i f_i(s_i(\lambda_i), \lambda_i) \quad (13)$$
$$\text{s.t.} \quad \sum_i \lambda_i = \lambda. \quad (14)$$

When $\mathcal{F}_i = E[T_i]$, we call it delay optimal load balancing.

Cost-aware optimal load balancing: The dispatcher routes arrivals so as to achieve social optimum in terms of cost-aware performance metric $\mathcal{M}_i$:

$$\min_{\lambda_i \geq 0} \quad \sum_i g_i(s_i(\lambda_i), \lambda_i) + \beta_i \lambda_i f_i(s_i(\lambda_i), \lambda_i) \quad (15)$$
$$\text{s.t.} \quad \sum_i \lambda_i = \lambda. \quad (16)$$

We call it energy-aware optimal load balancing in the case of energy-aware speed scaling.

The end users as a whole care about problem (13)-(14) and the servers/end users as a whole care about problem (15)-(16). We intend to characterize the LBSS equilibrium with respect to them, as well as proposing distributed algorithms to achieve the corresponding equilibrium or optima. Again, the general problems (13)-(14) and (15)-(16) may be highly nontrivial, depending on the performance curve f_i under speed scaling. In the remainder of this paper, we will focus on load balancing with energy-aware speed scaling in processor sharing systems with performance metric (2) with power function (3), as a concrete system to study the interaction between load balancing and speed scaling. We will leave the general problem to future work.

V. Load-Balancing-Speed-Scaling Interaction in Processor Sharing Systems

In this section, we consider energy-aware speed scaling in processor sharing (PS) systems with performance metric (2) and power function (3). While general speed scaling policies can be taken at a server, we focus on *gated-static speed scaling*, in which the server has a zero speed when there is no job and otherwise runs at a constant speed that balances the response time and energy usage; see, e.g, [9], [10]. Gated-static speed scaling is the simplest *nontrivial* speed scaling. It requires minimal hardware to support. For example, a CMOS chip may set a constant clock speed but AND it with the gating signal to set the speed to 0 when there is no job [10]. The gated static speed scaling captures some essence of dynamic speed scaling while admits more tractable analysis.

As mentioned in Section IV, when $\alpha_i < 2$, the problem under gated static speed scaling may become hard problem of minimizing a concave objective function. We thus focus on the system with $\alpha_i \geq 2$, in order to obtain a clean characterization to gain insights. Power functions with $\alpha_i \geq 2$ is also practically important, as in the server with a power function with $\alpha_i \geq 2$ energy cost is usually the driving force in deciding on server speed while in the server with a power function with $\alpha_i < 2$ traditional performance metric is the driving force. Besides, the results obtained for gated static speed scaling with $\alpha_i \geq 2$ are expected to carry over to static provisioning with $\alpha_i \geq 1$, in which the server runs at a constant static speed that is chosen based on workload to balance the response time and energy usage. Static provisioning is the simplest form of speed scaling, and is a model often used in energy-aware capacity provisioning in data centers.

A. Energy-oblivious load balancing

Under PS scheduling, the mean response time at server i takes the form:

$$f_i(s_i, \lambda_i) = \frac{1}{s_i - \lambda_i}. \tag{17}$$

Under gated static speed scaling, the energy cost is only incurred during the time when the server is busy. Note that the fraction of the time when the server is busy is λ_i/s_i. So, the server decides on speed s_i by solving the following optimization problem:

$$\min_{s_i > \lambda_i} \qquad \beta_i \frac{\lambda_i}{s_i - \lambda_i} + \frac{\lambda_i}{s_i} P_i(s_i). \tag{18}$$

Thus, the speed scaling $s_i(\lambda_i)$ satisfies

$$-\frac{\bar{\beta}_i}{(s_i - \lambda_i)^2} + s_i^{\alpha_i - 2} = 0, \tag{19}$$

where $\bar{\beta}_i = \frac{\beta_i}{k_i(\alpha_i - 1)}$. By equation (19), we have

$$s_i'(\lambda_i) \;=\; \frac{2s_i(\lambda_i)}{\alpha_i s_i(\lambda_i) - (\alpha_i - 2)\lambda_i} > 0, \tag{20}$$

$$s_i''(\lambda_i) \;=\; \frac{(2\alpha_i - 4)(s_i(\lambda_i) - \lambda_i s_i'(\lambda_i))}{(\alpha_i s_i(\lambda_i) - (\alpha_i - 2)\lambda_i)^2} \geq 0, \tag{21}$$

where the second inequality follows from the fact that $s_i'(\lambda_i) \leq 1$, and moreover, $s_i'(\lambda_i) = 1$ and $s_i''(\lambda_i) = 0$ if and only if $\alpha_i = 2$. Hence, speed scaling $s_i(\lambda_i)$ is a strictly increasing, convex function of λ_i. Further,

$$f_i(s_i(\lambda_i), \lambda_i) = \frac{1}{s_i(\lambda_i) - \lambda_i} = \sqrt{\frac{(s_i(\lambda_i))^{\alpha_i - 2}}{\bar{\beta}_i}} \tag{22}$$

is also a strictly increasing function of λ_i.

Corollary 3: There exists a unique LBSS equilibrium for processor sharing systems with gated-static speed scaling.

Proof: By Theorem 2, the LBSS equilibrium satisfies the optimality conditions for optimization problem:

$$\min_{\lambda_i} \qquad \sum_i \int \frac{1}{s_i(\lambda_i) - \lambda_i} d\lambda_i \tag{23}$$

$$\sum_i \lambda_i = \lambda. \tag{24}$$

Since $\frac{1}{s_i(\lambda_i) - \lambda_i}$ is strictly increasing in λ_i, the above optimization problem is strictly convex. The existence and uniqueness of LBSS equilibrium follows from the fact that problem (23)-(24) has a unique optimum [26]. ∎

Now, let us characterize the equilibrium. For each server i, define the "base" service rate $s_i^0 = s_i(0^+) = \bar{\beta}_i^{\frac{1}{\alpha_i}}$.[4] Without loss of generality, we assume that $s_1^0 \geq s_2^0 \geq \cdots \geq s_{|N|}^0$. For later convenience, we also assume that $s_{|N|+1}^0$=0.

Theorem 4: The set of servers that are used at the equilibrium is $N_e = \{1, 2, \cdots, n\}$, with a unique n that satisfies

$$\sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_n^0}) < \lambda \leq \sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_{n+1}^0}), \tag{25}$$

where

$$\tilde{f}_i(\lambda_i) = \frac{1}{s_i(\lambda_i) - \lambda_i}. \tag{26}$$

[4]For a function $f(x) : \mathcal{R} \mapsto \mathcal{R}$, $f(a^+)$ denotes the right hand limit $\lim_{x \to a^+} f(x)$.

Proof: By equilibrium condition (9), we have $\frac{1}{s_i^0} < \gamma$ if $i \in N_e$ and $\frac{1}{s_i^0} \geq \gamma$ otherwise. Further,

$$\lambda_i = s_i - \frac{1}{\gamma} > 0, \text{ if } \frac{1}{s_i^0} < \gamma \quad (27)$$

$$\lambda_i = 0, \text{ if } \frac{1}{s_i^0} \geq \gamma. \quad (28)$$

Since s_i^0 is decreasing in i, N_e takes the form of $\{1, 2, \cdots, n\}$.

Note that $\frac{1}{s_n^0} < \gamma \leq \frac{1}{s_{n+1}^0}$, and $\tilde{f}_i(\lambda_i)$ is an increasing function. So,

$$\sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_n^0}) < \sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\gamma) \leq \sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_{n+1}^0}),$$

i.e.,

$$\sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_n^0}) < \sum_{i=1}^{n} \lambda_i = \lambda \leq \sum_{i=1}^{n} (\tilde{f}_i)^{-1}(\frac{1}{s_{n+1}^0}).$$

The uniqueness of n follows from the fact that the LBSS equilibrium is unique. ■

We see that the LBSS equilibrium has a water-filling structure. If we see load balancing as a selfish routing problem [27], the arrivals will aggressively occupy fast servers with low delay first.

1) Distributed load balancing algorithm: The (convex) optimization problem characterization of the LBSS equilibrium also suggests a distributed algorithm to achieve the equilibrium.

At k-th iteration:

- Each server i estimates the arrival rate λ_i, and adjusts its speed s_i, according to

$$s_i(k) = s_i(\lambda_i(k)). \quad (29)$$

- The dispatcher measures delay $t_i(k) = \frac{1}{s_i(k) - \lambda_i(k)}$ experienced at each server i. Denote by $E[t(k)]$ the minimal $\bar{t}(k)$ at step k such that $\bar{t}(k) = \frac{1}{|\bar{N}(k)|} \sum_{i \in \bar{N}(k)} t_i(k)$ with $\bar{N}(k) := \{i | \lambda_i(k) > 0 \text{ or } t_i(k) \leq \bar{t}(k), i \in N\}$.[5] The dispatcher adjusts λ_i to each server i, according to

$$\lambda_i(k+1) = [\lambda_i(k) - \varepsilon(t_i(k) - E[t(k)])]^+. \quad (30)$$

  where ε is a positive stepsize, and '+' denotes the projection onto $\mathcal{R}^+$, the set of nonnegative real numbers.

When ε is small enough, the above algorithm converges. Let $\delta_i(k) = \lambda_i(k+1) - \lambda_i(k)$. It is easy to verify that

$$\sum_i \delta_i(k) = 0, \quad (31)$$

$$\sum_i \delta_i(k) t_i(k) \leq 0. \quad (32)$$

We see that $\sum_i \delta_i(k) t_i(k) = 0$ only if $\delta_i(k) = 0$, which requires $t_i = \bar{t}$, or, $\lambda_i = 0$ and $t_i > \bar{t}$.

[5] $\bar{t}$ and $\bar{N}$ can be determined in a recursive way as follows. In the beginning, let $\bar{N} = N$ and calculate $\bar{t} = \frac{1}{|\bar{N}|} \sum_{i \in \bar{N}} t_i(k)$, and then exclude from $\bar{N}$ those servers i such that $\lambda_i = 0$ and $t_i > \bar{t}$. Repeat the same procedure with the new sets $\bar{N}$, and when it stops we get $E[t]$.

The above algorithm actually follows the negative gradient direction of $\sum_i \int \frac{1}{s_i(\lambda_i) - \lambda_i} d\lambda_i$ subject to $\lambda_i = \lambda$ [26]. Any algorithms that follow a properly-chosen negative gradient direction would work, and (30) picks a specific gradient direction that will facilitate the convergence analysis. We skip the convergence proof for brevity.

B. Delay optimal load balancing

In this subsection, we study delay optimal load balancing design:

$$\min_{\lambda_i \geq 0} \quad \sum_i \frac{\lambda_i}{s_i(\lambda_i) - \lambda_i} \quad (33)$$

$$\text{s.t.} \quad \sum_i \lambda_i = \lambda, \quad (34)$$

and characterize the LBSS equilibrium with respect to it.

By equation (19),

$$\frac{\lambda_i}{s_i(\lambda_i) - \lambda_i} = \sqrt{\frac{1}{\bar{\beta}_i}} s_i^{\frac{\alpha_i}{2}} - 1, \quad (35)$$

which is strictly increasing and convex in s_i. Note that $s_i(\lambda_i)$ is increasing and convex. It follows that $\frac{\lambda_i}{s_i(\lambda_i) - \lambda_i}$ is a strictly convex function of λ_i.[6] So, problem (33)-(34) is strictly convex, and has a unique optimum. Denote the optimum by $(\lambda_i^*)_{i \in N}$. There exists a unique $\gamma^* > 0$, such that the optimality condition can be written as [26]

$$(\frac{s_i(\lambda_i^*) - \lambda_i^* s_i'(\lambda_i^*)}{(s_i(\lambda_i^*) - \lambda_i^*)^2} - \gamma^*)(\bar{\lambda}_i - \lambda_i^*) \geq 0, \ \forall \ \bar{\lambda}_i \geq 0, \quad (36)$$

$$\sum_{i \in N} \lambda_i^* = \lambda. \quad (37)$$

Theorem 5: The set of servers that are used at the optimum is $N_o = \{1, 2, \cdots, n^*\}$, with a unique n^* that satisfies

$$\sum_{i=1}^{n^*} (\hat{f}_i)^{-1}(\frac{1}{s_{n^*}^0}) < \lambda \leq \sum_{i=1}^{n^*} (\hat{f}_i)^{-1}(\frac{1}{s_{n^*+1}^0})\}, \quad (38)$$

where

$$\hat{f}_i(\lambda_i) = \frac{s_i(\lambda_i) - \lambda_i s_i'(\lambda_i)}{(s_i(\lambda_i) - \lambda_i)^2}. \quad (39)$$

Moreover, $\gamma^* \geq \gamma$ and $n^* \geq n$.

Proof: Note that $\hat{f}_i(\lambda_i)$ is an increasing function of λ_i, and $\hat{f}_i(0) = \frac{1}{s_i^0}$. The first part of the theorem follows the same proof as in Theorem 4.

For the second part of the theorem. Note that $s_i'(\lambda_i) \leq 1$ by equation (20). Thus, $\hat{f}_i(\lambda_i) \geq \tilde{f}_i(\lambda_i)$. If $\gamma^* < \gamma$, then $n^* \leq n$ and

$$\sum_{i=1}^{n^*} (\hat{f}_i)^{-1}(\gamma^*) < \sum_{i=1}^{n^*} (\tilde{f}_i)^{-1}(\gamma^*) \leq \sum_{i=1}^{n^*} (\tilde{f}_i)^{-1}(\gamma) \leq \lambda.$$

This contradicts $\sum_{i=1}^{n^*} (\hat{f}_i)^{-1}(\gamma^*) = \sum_{i=1}^{n^*} \lambda_i^* = \lambda$. So, $\gamma^* \geq \gamma$, and $n^* \geq n$ follows. ■

[6] Note that, when $\alpha_i = 2$, $\frac{\lambda_i}{s_i(\lambda_i) - \lambda_i}$ is not strictly convex but linear in λ_i. But this would not change the uniqueness of the optimum.

1) Distributed load balancing algorithm: The delay optimal load balancing is a convex problem. We can apply similar distributed algorithm to algorithm (29)-(30), to guide the optimal load balancing design.

At k-th iteration:

- Each server i estimates the arrival rate λ_i, and adjusts its speed s_i, according to

$$s_i(k) = s_i(\lambda_i(k)). \tag{40}$$

- The dispatcher measures delay $t_i(k) = \frac{1}{s_i(k)-\lambda_i(k)}$ experienced at each server i, and estimates $\hat{f}_i$, according to

$$\hat{f}_i(k) = \hat{f}_i(\lambda_i(k)) = \frac{\alpha_i \lambda_i(k)(t_i(k))^2 + \alpha_i t_i(k)}{2\lambda_i(k)t_i(k) + \alpha_i}. \tag{41}$$

  Denote by $E[\hat{f}(k)]$ the minimal $\bar{\hat{f}}(k)$ at step k such that $\bar{\hat{f}}(k) = \frac{1}{|\bar{N}(k)|}\sum_{i\in\bar{N}(k)} \hat{f}_i(k)$ with $\bar{N}(k) := \{i|\lambda_i(k) > 0 \text{ or } \hat{f}_i(k) \leq \bar{\hat{f}}(k), i \in N\}$. The dispatcher adjusts λ_i to each server i, according to

$$\lambda_i(k+1) = [\lambda_i(k) - \varepsilon(\hat{f}_i(k) - E[\hat{f}(k)])]^+. \tag{42}$$

  where ε is a positive stepsize, and '+' denotes the projection onto $\mathcal{R}^+$, the set of nonnegative real numbers.

Note that delay optimal load balancing algorithm (40)-(42) is more complicated than the simple, energy-oblivious load balancing algorithm (29)-(30). It requires to estimate $\hat{f}_i$. In addition, it requires the dispatcher to know the servers' power function characteristic parameters α_i and k_i.

2) Efficiency loss in delay at the LBSS equilibrium: Define the social cost in delay:

$$C = \sum_i \frac{\lambda_i}{s_i(\lambda_i) - \lambda_i}, \tag{43}$$

we now characterize the inefficiency in delay at the LBSS equilibrium.

Lemma 6: Let $\alpha = \max_i \alpha_i$. Then,

$$\gamma \leq \gamma^* \leq \frac{\alpha}{2}\gamma. \tag{44}$$

Proof: The first inequality has been proved in Theorem 5. It remains to prove the second one.

By equation (35), $\hat{f}_i$ can be written as

$$\hat{f}_i(\lambda_i) = \frac{\alpha_i}{2}\sqrt{\frac{s_i^{\alpha_i-2}}{\bar{\beta}_i}}\; s_i'. \tag{45}$$

Note that $s_i'(\lambda_i)$ is increasing. Thus, $s'(\lambda_i) \geq \frac{2}{\alpha_i}$ by equation (20). Combining with $s'(\lambda_i) \leq 1$, we get

$$\tilde{f}_i(\lambda_i) \leq \hat{f}_i(\lambda_i) \leq \frac{\alpha_i}{2}\tilde{f}_i(\lambda_i) \leq \frac{\alpha}{2}\tilde{f}_i(\lambda_i).$$

If $\gamma^* > \frac{\alpha}{2}\gamma$, then

$$(\hat{f}_i)^{-1}(\gamma^*) \geq (\tilde{f}_i)^{-1}(\frac{2}{\alpha_i}\gamma^*) > (\tilde{f}_i)^{-1}(\gamma).$$

Thus,

$$\sum_{i=1}^{n^*}(\hat{f}_i)^{-1}(\gamma^*) > \sum_{i=1}^{n}(\tilde{f}_i)^{-1}(\gamma) = \lambda.$$

This contradicts the fact that $\sum_{i=1}^{n^*}(\hat{f}_i)^{-1}(\gamma^*) = \lambda$ (also note that $n^* \geq n$). So, $\gamma^* \leq \frac{\alpha}{2}\gamma$. ■

Theorem 7: Denote the social cost in delay at the LBSS equilibrium by C^e and the optimal cost by C^o. Then,

$$\frac{C^e}{C^o} \leq \frac{\alpha}{2}. \tag{46}$$

Proof: The social cost at the LBSS equilibrium is

$$C^e = \lambda\gamma. \tag{47}$$

When $\lambda_i^* > 0$, by equations (22), (45) and (44), we have

$$\frac{1}{s_i(\lambda_i^*) - \lambda_i^*} = \sqrt{\frac{s_i^{\alpha_i-2}}{\bar{\beta}_i}} = \frac{2\gamma^*}{\alpha_i s_i'} \geq \frac{2\gamma^*}{\alpha_i} \geq \frac{2\gamma}{\alpha}. \tag{48}$$

So,

$$C^o = \sum_i \frac{\lambda_i^*}{s_i(\lambda_i^*) - \lambda_i^*} \geq \frac{2\gamma}{\alpha}\sum_i \lambda_i^* = \frac{2\lambda\gamma}{\alpha}. \tag{49}$$

Thus,

$$\frac{C^e}{C^o} \leq \frac{\alpha}{2}. \tag{50}$$

■

We see that the degree of inefficiency in delay at the LBSS equilibrium depends only on the order α_i of the power functions. For example, if $\alpha_i = 2$, the LBSS equilibrium achieves the social optimum. As α is a constant independent of the number $|N|$ of the servers in the system, this result is very different from the efficiency loss of the usual load balancing (with fixed server speeds), which scales with $|N|$, see, e.g., [22]. Also, note that $\frac{\alpha}{2}$ can be seen as a measure of heterogeneity in power functions. We can thus say that the degree of inefficiency at the LBSS equilibrium is bounded by the heterogeneity of the system. As the power function can usually be well approximated as a low-order polynomial function, the above result suggests "benign" interaction between energy-oblivious load balancing and power-aware speed scaling, in terms of delay. As energy-oblivious load balancing is already employed in practice and simple to implement, we may need not change it as it does not incur a large penalty in delay.

C. Energy-aware optimal load balancing

In this subsection, we study energy-aware optimal load balancing design:

$$\min_{\lambda_i, s_i} \quad \sum_i \beta_i \frac{\lambda_i}{s_i - \lambda_i} + \frac{\lambda_i P_i(s_i)}{s_i} \tag{51}$$

$$\text{s.t.} \quad \sum_i \lambda_i = \lambda, \tag{52}$$

and characterize the LBSS equilibrium with respect to it.

By speed scaling (i.e., solving for s_i first), the above problem reduces to:

$$\min_{\lambda_i} \quad \sum_i h_i(\lambda_i) \tag{53}$$

$$\text{s.t.} \quad \sum_i \lambda_i = \lambda, \tag{54}$$

where

$$h_i(\lambda_i) = \beta_i \frac{\lambda_i}{s_i(\lambda_i) - \lambda_i} + \frac{\lambda_i P_i(s_i(\lambda_i))}{s_i(\lambda_i)}. \quad (55)$$

Note that

$$\begin{aligned} h_i'(\lambda_i) &= \frac{\beta_i s_i(\lambda_i)}{(s_i(\lambda_i) - \lambda_i)^2} + k_i(s_i(\lambda_i))^{\alpha_i - 1} \\ &= \frac{\alpha_i \beta_i}{\alpha_i - 1} \frac{s_i(\lambda_i)}{(s_i(\lambda_i) - \lambda_i)^2}, \quad (56) \\ h_i''(\lambda_i) &= \frac{\alpha_i \beta_i}{\alpha_i - 1} \frac{2s_i(\lambda_i) - (s_i(\lambda_i) + \lambda_i)s_i'(\lambda_i)}{(s_i(\lambda_i) - \lambda_i)^3}. \quad (57) \end{aligned}$$

We see that $h_i' > 0$ and $h_i'' > 0$, and thus $h_i(\lambda_i)$ is strictly increasing and convex. So, problem (53)-(54) is a strictly convex problem, and has a unique optimum. Denote the optimum by $(\lambda_i^+)_{i \in N}$. There exists a unique $\gamma^+ > 0$, such that the optimality condition can be written as [26]

$$(h_i'(\lambda_i^+) - \gamma^+)(\bar{\lambda}_i - \lambda_i^+) \geq 0, \ \forall \ \bar{\lambda}_i \geq 0, \quad (58)$$

$$\sum_{i \in N} \lambda_i^+ = \lambda. \quad (59)$$

Note that $h_i'(\lambda_i)$ is strictly increasing, and

$$h_i'(\lambda_i) \geq \frac{\alpha_i \beta_i}{\alpha_i - 1} \hat{f}_i(\lambda_i) \geq \frac{\alpha_i \beta_i}{\alpha_i - 1} \tilde{f}_i(\lambda_i). \quad (60)$$

Let $d_i^0 = \frac{1}{h_i'(0)} = \frac{\alpha_i - 1}{\alpha_i \beta_i} s_i^0$. We can define a permutation $\pi : \{1, 2, \cdots, |N|\} \mapsto \{1, 2, \cdots, |N|\}$, such that d_i^0 is in decreasing order under π. We have the following characterization of the optimum.

Theorem 8: The set of servers that are used at the optimum is $N_s = \{\pi^{-1}(1), \pi^{-1}(2), \cdots, \pi^{-1}(m^+)\}$, with a unique m^+ that satisfies

$$\sum_{i=1}^{m^+} (h'_{\pi^{-1}(i)})^{-1} \left(\frac{1}{d^0_{\pi^{-1}(m^+)}}\right) < \lambda \leq \sum_{i=1}^{m^+} (h'_{\pi^{-1}(i)})^{-1} \left(\frac{1}{d^0_{\pi^{-1}(m^+ + 1)}}\right).$$

Proof: It follows the same proof as in Theorem 4. We skip it for brevity. ■

We see that the energy-aware optimal load balancing has a similar water-filling effect, and the arrivals will occupy servers with low marginal cost in energy-aware metric first. As a result, the jobs will be consolidated into a subset of servers that have low energy-aware cost.

1) Distributed load balancing algorithm: The energy-aware optimal load balancing is a convex problem. Again, we can apply similar distributed algorithm to algorithm (29)-(30), to guide the optimal load balancing design.

At k-th iteration:

- Each server i estimates the arrival rate λ_i, and adjusts its speed s_i, according to

$$s_i(k) = s_i(\lambda_i(k)). \quad (61)$$

- The dispatcher measures delay $t_i(k) = \frac{1}{s_i(k) - \lambda_i(k)}$ experienced at each server i, and estimates h_i', according to

$$h_i'(k) = h_i'(\lambda_i(k)) = \frac{\alpha_i \beta_i}{\alpha_i - 1} (\lambda_i(k)(t_i(k))^2 + t_i(k)). \quad (62)$$

Denote by $E[h'(k)]$ the minimal $\bar{h}'(k)$ at step k such that $\bar{h}'(k) = \frac{1}{|\bar{N}(k)|} \sum_{i \in \bar{N}(k)} h_i'(k)$ with $\bar{N}(k) := \{i | \lambda_i(k) > 0 \text{ or } h_i'(k) \leq \bar{h}'(k), i \in N\}$. The dispatcher adjusts λ_i to each server i, according to

$$\lambda_i(k+1) = [\lambda_i(k) - \varepsilon(h_i'(k) - E[h'(k)])]^+. \quad (63)$$

where ε is a positive stepsize, and '+' denotes the projection onto $\mathcal{R}^+$, the set of nonnegative real numbers.

Again, energy aware optimal load balancing algorithm (61)-(63) is more complicated than energy-oblivious load balancing algorithm (29)-(30). In addition to the servers' power function characteristic parameters, the dispatcher requires to know their weights β_i.

2) Efficiency loss in energy-aware performance metric at the LBSS equilibrium: Define the social cost in energy-aware performance metric $\mathcal{M}_i$:

$$D = \sum_i \beta_i \frac{\lambda_i}{s_i - \lambda_i} + \frac{\lambda_i P_i(s_i)}{s_i} = \sum_i h_i(\lambda_i). \quad (64)$$

We now characterize the inefficiency in energy-aware performance metric at the LBSS equilibrium.

It is complicated to characterize the efficiency loss for the system with arbitrary power functions and loads. Here we give a partial characterization, focusing on the case with power functions of the same order, i.e., $P_i(s_i) = k_i s_i^\alpha$ for all servers, and in heavy traffic, i.e., $\lambda \gg 1$. We leave a complete characterization of the efficiency loss to future work.

The case with power functions of the same order models a system that employs similar servers but with different scaling factors and weights. Heavy traffic regime is of significant interest, as the inefficiency of load-balancing-speed-scaling interaction is intuitively worst under heavy traffic.

Theorem 9: Assume that $\alpha = 2$. Denote the energy-aware social cost at the LBSS equilibrium by D^e and the optimal cost by D^o. Under the aforementioned conditions, we have

$$\frac{D^e}{D^o} \leq \frac{\max_i k_i}{\min_i k_i} |N|. \quad (65)$$

Proof: When $\alpha = 2$, at the LBSS equilibrium $(\lambda_i)_{i \in N}$, the arrivals will be routed to the server i^* that has the maximal $\bar{\beta}_i$ value.[7] Under heavy traffic, the energy-aware social cost at the LBSS equilibrium is

$$D^e \approx k_{i^*} \lambda^2 \leq \max_i k_i \lambda^2.$$

At the social optimum $(\lambda_i^+)_{i \in N}$, $\lambda_i^+ \approx \frac{1/k_i}{\sum_j 1/k_j} \lambda$. The optimal social cost is

$$D^0 \approx \sum_i k_i (\lambda_i^+)^2 \approx \frac{\lambda^2}{\sum_i 1/k_i} \geq \frac{\min_i k_i}{|N|} \lambda^2.$$

Thus,

$$\frac{D^e}{D^o} \leq \frac{\max_i k_i}{\min_i k_i} |N|. \quad (66)$$

[7] There may exist multiple servers that have the maximal $\bar{\beta}_i$ value. But it is reasonable to expect that the number of such servers is bounded by a constant that does not scale with the total number of the servers in the system. For simplicity of presentation, we assume that there is only one server that has the maximal $\bar{\beta}_i$ value. This only brings in a constant factor to the bound on efficiency loss, if there are multiple such servers.

We see that when $\alpha = 2$, the degree of inefficiency at the LBSS equilibrium scales with the number of servers in the system. This happens because the energy-oblivious load balancing uses only the server with the largest base rate, which incurs a huge energy cost at this server, while the energy-aware optimal load balancing will spread load across all servers, which leads to much smaller energy cost at the servers. This suggests that we should do energy-aware load balancing if the energy consumption is a main concern.

Lemma 10: Assume $\alpha > 2$. Define $\zeta_i = \alpha k_i \bar{\beta}_i^{\frac{\alpha-1}{\alpha-2}}$ for each server i,. Then,

$$\min_i \zeta_i \gamma^{\frac{2\alpha-2}{\alpha-2}} \leq \gamma^+ \leq \max_i \zeta_i \gamma^{\frac{2\alpha-2}{\alpha-2}}. \quad (67)$$

Proof: By equation (56), h_i' can be written as

$$h_i'(\lambda_i) = \alpha k_i (s_i(\lambda_i))^{\alpha-1} = \zeta_i (\tilde{f}_i(\lambda_i))^{\frac{2\alpha-2}{\alpha-2}}. \quad (68)$$

If $\gamma^+ < \min_i \zeta_i \gamma^{\frac{2\alpha-2}{\alpha-2}}$, then

$$(h_i')^{-1}(\gamma^+) < (\tilde{f}_i)^{-1}(\gamma).$$

Thus,

$$\sum_i (h_i')^{-1}(\gamma^+) < \sum_i (\tilde{f}_i)^{-1}(\gamma) = \lambda.$$

This contradicts the fact that $\sum_i (h_i')^{-1}(\gamma^+) = \lambda$. So, $\gamma^+ \geq \min_i \zeta_i \gamma^{\frac{2\alpha-2}{\alpha-2}}$. The second inequality can be proved similarly. ■

Theorem 11: Assume $\alpha > 2$. Denote the energy-aware social cost at the LBSS equilibrium by D^e and the optimal cost by D^o. Under the aforementioned conditions, we have

$$\frac{D^e}{D^o} \leq \left(\frac{\max_i \zeta_i}{\min_i \zeta_i}\right)^{\frac{\alpha}{\alpha-1}}. \quad (69)$$

Proof: Under heavy traffic, $\lambda_i \gg 1$. By Lemma 1 in [9], we have the following approximation for speed scaling under heavy traffic:

$$s_i(\lambda_i) \approx \lambda_i + \sqrt{\frac{\bar{\beta}_i}{\lambda_i^{\alpha-2}}} \approx \lambda_i.$$

Thus,

$$\beta_i \frac{\lambda_i}{s_i - \lambda_i} + \frac{\lambda_i P_i(s_i)}{s_i} \approx \beta_i \frac{\lambda_i^{\frac{\alpha}{2}}}{\sqrt{\bar{\beta}_i}} + k_i \lambda_i^{\alpha} \approx k_i \lambda_i^{\alpha}.$$

Note that, at the LBSS equilibrium $(\lambda_i)_{i\in N}$,

$$\gamma = \sqrt{\frac{s_i^{\alpha-2}}{\bar{\beta}_i}} \approx \sqrt{\frac{\lambda_i^{\alpha-2}}{\bar{\beta}_i}}.$$

The energy-aware social cost at the LBSS equilibrium is

$$D^e \approx \sum_i k_i (\bar{\beta}_i \gamma^2)^{\frac{\alpha}{\alpha-2}} \leq \sum_i k_i \bar{\beta}_i^{\frac{\alpha}{\alpha-2}} \left(\frac{\gamma^+}{\min_j \zeta_j}\right)^{\frac{\alpha}{\alpha-1}}, \quad (70)$$

where the inequality follows from (67).

At the social optimum $(\lambda_i^+)_{i\in N}$,

$$\gamma^+ = \alpha k_i s_i^{\alpha-1} \approx \alpha k_i (\lambda_i^+)^{\alpha-1}. \quad (71)$$

The optimal social cost is

$$D^o \approx \sum_i k_i \left(\frac{\gamma^+}{k_i \alpha}\right)^{\frac{\alpha}{\alpha-1}}. \quad (72)$$

Thus,

$$\begin{aligned}
\frac{D^e}{D^o} &\leq \frac{\sum_i k_i \bar{\beta}_i^{\frac{\alpha}{\alpha-2}} \left(\frac{\gamma^+}{\min_j \zeta_j}\right)^{\frac{\alpha}{\alpha-1}}}{\sum_i k_i \left(\frac{\gamma^+}{k_i \alpha}\right)^{\frac{\alpha}{\alpha-1}}} \\
&\leq \max_i \frac{k_i \bar{\beta}_i^{\frac{\alpha}{\alpha-2}} \left(\frac{\gamma^+}{\min_j \zeta_j}\right)^{\frac{\alpha}{\alpha-1}}}{k_i \left(\frac{\gamma^+}{k_i \alpha}\right)^{\frac{\alpha}{\alpha-1}}} \\
&= \max_i \left(\frac{\zeta_i}{\min_j \zeta_j}\right)^{\frac{\alpha}{\alpha-1}} \\
&\leq \left(\frac{\max_i \zeta_i}{\min_j \zeta_j}\right)^{\frac{\alpha}{\alpha-1}}. \quad (73)
\end{aligned}$$

■

We see that when $\alpha > 2$, the degree of inefficiency at the LBSS equilibrium depends only on the degree of heterogeneity $\frac{\max_i \zeta_i}{\min_j \zeta_j}$ in the system but not the number of servers $|N|$. If the degree of heterogeneity in the system is small, energy-oblivious load balancing interacts benignly with speed scaling, in terms of the energy-aware cost. In this situation, we may do not need complicated energy-aware load balancing, i.e., we can decouple the design of load balancing from speed scaling. Otherwise, we must do energy-aware optimal load balancing if energy consumption is a main concern.

VI. Numerical Examples

In this section, we provide numerical examples to complement the analysis in previous sections, mainly focusing on evaluating the distributed load balancing algorithms as our other results on the LBSS equilibrium, delay-optimal load balancing and energy-aware optimal load balancing are analytical results. We consider a system with 10 servers with speed scaling. Half of the servers have a power function of the form $P_i(s_i) = k_i s_i^{\frac{5}{2}}$ and the other half have a power function of the form $P_i(s_i) = k_i s_i^3$. The total load is normalized to be $\lambda = 10$, and the values for parameter k_i and β_i used to obtain numerical results are randomly drawn from $[1, 10]$ and $[5, 15]$, respectively.

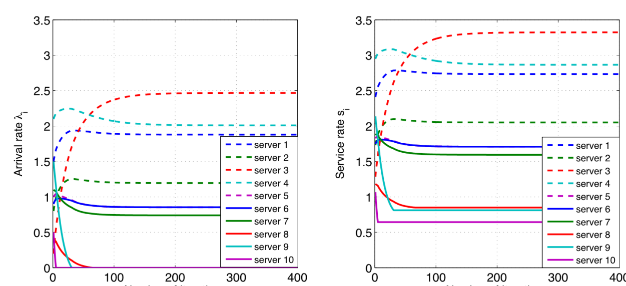

Fig. 2. The arrival rate and service rate evolution of energy-oblivious load balancing.

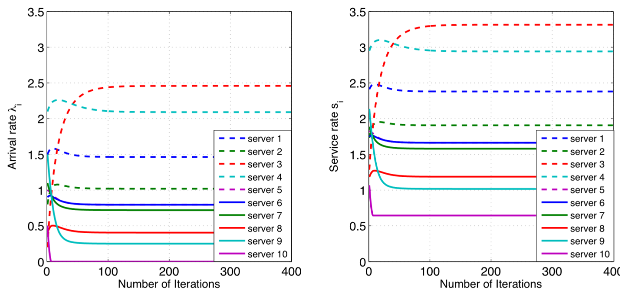

Fig. 3. The arrival rate and service rate evolution of delay optimal load balancing.

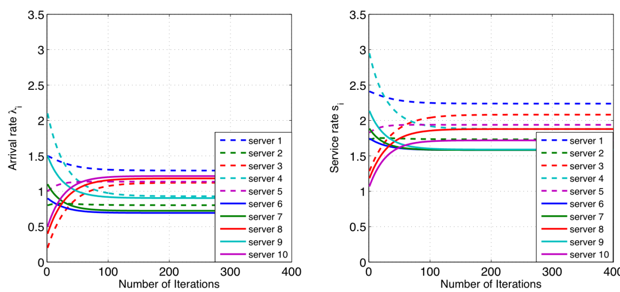

Fig. 4. The arrival rate and service rate evolution of energy-aware optimal load balancing.

Figures 2, 3 and 4 show the evolution of the arrival rate and service rate with stepsize $\gamma = 0.2$ for energy-oblivious load balancing, delay optimal load balancing and energy-aware optimal load balancing, respectively. We see that the arrival rates and service rates approach the corresponding equilibrium or optimum quickly. The numerical results confirm previous analysis and intuitions. As we go from energy-oblivious load balancing to delay optimal load balancing, the load is spread more across the servers, which is driven by minimizing the social cost in delay. We also see that the changes in the arrival rate and service rate are not severe, which intuitively confirms Theorem 7 that gives a small bound on efficiency loss at the LBSS equilibrium. As we move to energy-aware optimal load balancing, the load becomes more evenly distributed. This is driven by minimizing the energy-aware social cost, and an uneven load distribution will lead to uneven service rate distribution, which may result in large cost in energy at the server(s) with large speed. We also see large changes in the arrival rate and service rate. This implies a large degree of inefficiency at the LBSS equilibrium, which intuitively confirms Theorem 11 even though it is a characterization for the system with power functions of the same order.

In order to study the impact of different choices of the stepsize on the convergence of the algorithms, we have run simulations with different stepsizes. We found that the smaller the stepsize, the slower the convergence, and the larger the stepsize, the faster the convergence but the system may only approach to within a certain neighborhood of the equilibrium, which is a general characteristic of any gradient based method. In practice, the dispatcher can first choose large stepsizes to ensure fast convergence, and subsequently reduce the stepsizes once the price starts oscillating around some mean value.

VII. Conclusion

We have studied the interaction between load balancing and speed scaling. We characterize the equilibrium resulting from the load balancing and speed scaling interaction, and introduce two optimal load balancing designs, in terms of traditional performance metric and cost-aware (in particular, energy-aware) performance metric respectively. We study in detail the load-balancing-speed-scaling equilibrium and the optimal load balancing designs in processor sharing systems with gated-static speed scaling, and propose distributed load balancing algorithms to achieve the corresponding equilibrium and optima. Especially, we characterize the degree of inefficiency at the load-balancing-speed-scaling equilibrium in terms of delay as well as energy-aware metric, and show that the degree of inefficiency is mostly bounded by the heterogeneity of the system, but independent of the number of the servers. These results provide insights in understanding the interaction of load balancing with speed scaling and guiding new designs.

Further research stemming out of this paper includes the following. We are characterizing the efficiency loss in energy-aware metric at the load-balancing-speed-scaling equilibrium for the system with power functions of different polynomial orders. We are also studying the load balancing and speed scaling interaction in the processor sharing system with general power functions (e.g., nonconvex, discontinuous, with possibly a discrete set of allowable speeds), as well as in the system with other scheduling policies such as Shortest Remaining Processing Time (SRPT). We will further study other speed scaling policies and their impact on the design and performance of load balancing. Finally, we will go beyond energy-aware speed scaling, and study other types of speed scaling behaviors and their interaction with load balancing in, e.g., date centers or call centers.

References

[1] O. S. Unsal and I. Koren. System-level power-aware deisgn techniques in real-time systems. *Proc. IEEE*, 97(3):1055–1069, 2003.

[2] S. Kaxiras and M. Martonosi. *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool, 2008.

[3] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[4] L. Yuan and G. Qu. Analysis of energy reduction on dynamic voltage scaling-enabled systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 24(12):1827–1837, 2005.

[5] Y. Zhu and F. Mueller. Feedback edf scheduling of real-time tasks exploiting dynamic voltage scaling. *Real Time Systems*, 31:33–63, 2005.

[6] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, 2007.

[7] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proc. ISLPED*, 2007.

[8] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. ACM-SIAM SODA*, 2009.

[9] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proceedings of IEEE Infocom*, 2009.

[10] L. L. Andrew, M. Lin, and A. Wierman. Optimality, fairness, and robustness in speed scaling designs. In *Proceedings of ACM Sigmetrics*, 2010.

[11] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow times. In *Proc. ACM-SIAM SODA*, 2007.
[12] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.
[13] J. M. George and J. M. Harrison. Dynamic control of a queue with adjustable service rate. *Operations Research*, 49(5):720–731, 2001.
[14] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. In *Scandinavian Worksh. Alg. Theory*, 2004.
[15] J. R. Bradley. Optimal control of a dual service rate m/m/1 production-inventory model. *European Journal of Operations Research*, 161(3):812–837, 2005.
[16] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *Lecture Notes in Computer Science*, 3884:621–633, 2006.
[17] D. P. Bunde. Power-aware scheduling for makespan and flow. In *Proc. ACM Symp. Parallel Alg. and Arch*, 2006.
[18] S. Zhang and K. S. Catha. Approximation algorithm for the temperature-aware scheduling problem. In *Proceedings of IEEE Conference on Computer Aided Design*, 2007.
[19] N. Bansal, H.-L. Chan, T.-W. Lam, and L.-K. Lee. Scheduling for speed bounded processors. In *Int. Colloq. Automata, Languages and Programming*, 2008.
[20] N. Bansal, H.-L. Chan, K. Pruhs, and D. Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. In *Int. Colloq. Automata, Languages and Programming*, 2009.
[21] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong. Speed scaling functions for flow time scheduling based on active job count. In *Proc. Euro. Symp. Alg.*, 2009.
[22] M. Haviv and T. Roughgarden. The price of anarchy in an exponential multi-server. *Operations Research Letters*, 35:421–426, 2007.
[23] T. Wu and D. Starobinski. On the price of anarchy in unbounded delay networks. In *Proc. of Game Theory for Comm. and Networks*, 2006.
[24] E. Altman, U. Ayesta, and B. J. Prabhu. Optimal load balancing in processor sharing systems. In *Proceedings of GameComm*, 2008.
[25] H. Chen, J. Marden, and A. Wierman. On the impact of heterogeneity and back-end scheduling in load balancing designs. In *Proceedings of IEEE Infocom*, 2009.
[26] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice Hall, 1989.
[27] N. Nissan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.